PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: SANG WON KANG ET AL
SERIAL NO.: 09/749,782
FILED: December 28, 2000
FOR: HIGH-SPEED SEARCH METHOD FOR LSP QUANTIZER USING SPLIT VQ AND FIXEX CODEBOOK OF G.729 SPEECH ENCODER

GROUP ART UNIT: 2654
EXAMINER: D. Storm
ATTY. REFERENCE: KANG3005/BEU

COMMISSIONER OF PATENTS
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

The below identified communication(s) or document(s) is(are) submitted in the above application or proceeding:

- ☐ Declaration
- ☐ Priority Document
- ☐ Formal Drawings
- ☐ Issue Fee Transmittal
- ☐ Check $_____
- ☐ Application Data Sheet
- ☒ **Exhibit I and Exhibit II**

☒ Please debit or credit **Deposit Account Number 02-0200** for any deficiency or surplus in connection with this communication. A duplicate copy of this sheet is provided for use by the Deposit Account Branch.

☐ Small Entity Status is claimed.

☐

23364
Customer Number

BACON & THOMAS, PLLC
625 SLATERS LANE - FOURTH FLOOR
ALEXANDRIA, VIRGINIA 22314
(703) 683-0500

DATE: March 11, 2005

Respectfully submitted,

Benjamin E. Urcia
Attorney for Applicant
Registration Number: 33,805

Exhibit I
Ser. No. 09/749,782
Art Unit 2654
Ext D. Storm

Speech Coder using Line Spectral Frequencies of Cascaded Second Order Predictors

by

Visala Namburu

Thesis submitted to the Faculty of
The Bradley Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

APPROVED

Dr. A. A. (Louis) Beex, Chairman

Dr. Brian D. Woerner

Dr. William T. Baumann

November 2001
Blacksburg, VA

KEYWORDS: Speech Coding, Linear Prediction, Cascaded Second Order Predictors, Line Spectral Frequencies, Vector Quantization.

In practice, a 256-point FFT is used to approximate the spectrum of $A(z)$ and $A_q(z)$.

Transparent quantization of LPC parameters means that the reconstructed speech produced quantized LSF is audibly indistinguishable from the original speech produced from the un-quantized LSF. It can be achieved using 32-34 bits/frame using scalar quantization (SQ) techniques, while vector quantization (VQ) using 24-26 bits/frame retains similar quality [22]. The VQ approaches include full-search VQ (FSVQ), split VQ (SVQ), multi-stage VQ (MSVQ), shape-gain VQ (SGVQ), and tree-search VQ (TSVQ). A FSVQ requires a large codebook to meet the transparency requirements, thus leading to an increase in the search complexity. However, using a structured codebook such as a tree-search codebook, split codebook and multi-stage codebook the search complexity can be reduced remarkably [11].

We describe the vector quantization, codebook structure and the *generalized Lloyd algorithm* for codebook design in Section 4.1. The basic concepts behind the SVQ and the SGVQ are described in Section 4.2 and Section 4.3 respectively.

4.1 Vector Quantization

A vector quantizer (VQ) quantizes a block of input data as a single vector, thus VQ is multidimensional, unlike the uni-dimensional scalar quantizer. A VQ produces less distortion than a scalar quantizer for the same number of bits [7]. VQ exploits linear and non-linear dependence among the vectors to be quantized. VQ allows different cell shapes, like hexagons, to fill the region $\Re^K$. This is unlike in SQ, where the region $\Re^K$ is filled with rectangular cells. For example, let $\aleph$ be a plane as shown in Figure 4-1(a). Figure 4-1(b) and Figure 4-1(c) depict the rectangular and the hexagonal partitions of $\aleph$ respectively. For the same number of quantization cells, hexagonal cell shapes result in a lower worst-case error for a statistical error criterion such as Euclidean distance than rectangular cells, provided the edge effects are negligible [7]. The advantage of VQ is that it allows different cell shapes like hexagons that fill the region $\Re^K$ more efficiently than the rectangular cells in allowed in SQ.

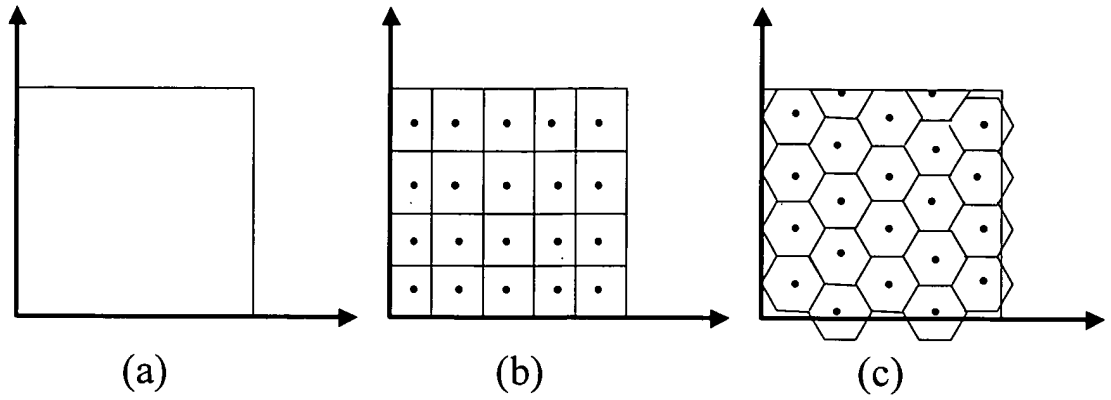

Figure 4-1 Two Dimensional Quantization a) Plane, ℵ b) Rectangular Partition c) Hexagonal Partition.

Let $\mathbf{y}$ be an input vector of length K defined as

$$\mathbf{y} = [y_0 \quad y_1 \quad \cdots \quad y_{K-1}]^T \tag{4-4}$$

VQ quantizes $\mathbf{y}$ to $\hat{\mathbf{y}}$,

$$\hat{\mathbf{y}} = Q(\mathbf{y}) \tag{4-5}$$

where $Q(\ )$ is the vector quantization operation.

The vector $\hat{\mathbf{y}}$ is chosen from a set of L code words $\mathbf{C} = \mathbf{c}_i$, $0 \le i \le L-1$ such that the nearest neighbor rule

$$d(\mathbf{y},\mathbf{c}_i) \le d(\mathbf{y},\mathbf{c}_k), \quad 0 \le i < k \le L-1, i \ne k \tag{4-6}$$

is satisfied. The parameter $d(\mathbf{x},\mathbf{z})$ denotes a distortion measure between $\mathbf{x}$ and $\mathbf{z}$, $\mathbf{x}$ and $\mathbf{z}$ being column vectors of length K each. The set of vectors $\mathbf{C}$, called the codebook, consists of L code vectors and is represented as

$$\mathbf{C} = [\mathbf{c}_0 \quad \mathbf{c}_1 \quad \cdots \quad \mathbf{c}_{L-1}] \tag{4-7}$$

where $\mathbf{c}_i$ is the i^{th} code vector of length K, defined as

$$\mathbf{c}_i = \begin{bmatrix} c_{0,i} & c_{1,i} & \cdots & c_{K-1,i} \end{bmatrix}^T \tag{4-8}$$

Thus, a vector quantizer Q of dimension $K \times L$ maps a vector in the K dimensional Euclidean subspace $\Re^K$, to the finite codebook $\mathbf{C}$ of size $K \times L$ and chooses the code vector that is closest to the input vector.

$$Q : \Re^K \rightarrow \mathbf{C} \tag{4-9}$$

The resolution rate r is the number of bits per vector component used to represent the input vector,

$$r = {(\log_2 L)}/{K} = {B}/{K} \tag{4-10}$$

where B is the number of bits needed to address the code words in $\mathbf{C}$. VQ achieves fractional values of resolution, as defined in (4-10), essential for low-bit rate applications [7].

A distortion measure $d(\mathbf{y}, \hat{\mathbf{y}})$ associated with quantizing any input vector $\mathbf{y}$ to $\hat{\mathbf{y}}$ can be used to evaluate the performance of a system. A quantizer is good if the average distortion is small. The most common distance measures are the squared error (SE) distance measure denoted by $d_{SE}(\mathbf{y}, \hat{\mathbf{y}})$, the mean-square error (MSE) distance measure denoted by $d_{MSE}(\mathbf{y}, \hat{\mathbf{y}})$, and the weighted mean square error ($WMSE$) distance measure denoted by $d_{WMSE}(\mathbf{y}, \hat{\mathbf{y}})$. These distance measures are respectively defined as follows

$$\begin{aligned} SE = d_{SE}(\mathbf{y}, \hat{\mathbf{y}}) &= (\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}}) \\ &= \sum_{k=1}^{K} (y_k - \hat{y}_k)^2 \end{aligned} \tag{4-11}$$

$$MSE = d_{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{K} (\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}}) \tag{4-12}$$

$$WMSE = d_{WMSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{K} (\mathbf{y} - \hat{\mathbf{y}})^T \mathbf{W} (\mathbf{y} - \hat{\mathbf{y}}) \tag{4-13}$$

where $\mathbf{W}$ is a symmetric and positive weighting matrix of size $K \times K$ [7], $\mathbf{y}$ and $\hat{\mathbf{y}}$ are column vectors each of the same length. The *WMSE* measure includes the *MSE* measure for $\mathbf{W} = \mathbf{I}$, the identity matrix. In speech and image compression applications, a matrix $\mathbf{W}$ that depends explicitly on the input vector $\mathbf{y}$ to be quantized, is chosen to obtain perceptually motivated distortion measures [7]. For example, let $\mathbf{W}(\mathbf{y})$ be $\|\mathbf{y}\|^{-2}\mathbf{I}$, where $\mathbf{I}$ is the identity matrix and $\|\mathbf{y}\| > 0$, then (4-13) turns out to be the ratio of noise energy to signal energy,

$$d_{WMSE}(\mathbf{y},\hat{\mathbf{y}}) = \frac{\|\mathbf{y}-\hat{\mathbf{y}}\|^2}{\|\mathbf{y}\|^2} \tag{4-14}$$

For a given noise energy $\|\mathbf{y}-\hat{\mathbf{y}}\|^2$, $d_{WMSE}(\mathbf{y},\hat{\mathbf{y}})$ is higher when $\mathbf{y}$ is small than when $\mathbf{y}$ is large.

4.1.1 Codebook Structure

The codebook $\mathbf{C}$ of dimension $K \times L$, is obtained by training with a large number of input vectors; these are also called the training vectors. The training data is partitioned into L Voronoi regions or cells R_i in the K dimensional subspace $\Re^K$ such that there exists no overlap region [7],

$$\bigcup_{i=1}^{L} R_i = \Re^K; \quad R_i \bigcap R_j = [\,], \; for \quad i \neq j \tag{4-15}$$

The centroid $\mathbf{c}_i$ of the Voronoi region R_i is the code word representing R_i. The Voronoi region R_i [10][17] associated with a code vector $\mathbf{c}_i$ is defined as

$$R_i = \left[\mathbf{y} \subset \Re^K : \|\mathbf{y}-\mathbf{c}_i\| < \|\mathbf{y}-\mathbf{c}_k\|; \quad 0 \le i \le L-1, i \neq k\right] \tag{4-16}$$

VQ quantizes $\mathbf{y}$ to $\mathbf{c}_i$ if $\mathbf{y} \in R_i$.

4.1.2 Codebook Initializations

A codebook can be initialized in different ways. The initialization of the codebook affects the overall performance of the resulting codebook. An initial codebook can be obtained by randomly

Chapter 2 Linear Prediction of Speech

In practical speech coding scenarios, speech is usually sampled at 8 kHz. Speech has at most four recognizable formant frequencies when it is sampled at 8 kHz. Thus, we attempt to model speech with an AR process of at least 8^{th} order. Most speech coding applications, use 10^{th} order AR filters. Speech is usually considered to be stationary within a 16 to 32 ms interval, leading to a corresponding analysis interval of 128 to 256 samples [5].

Linear Predictive Coding (LPC) is one of the analysis techniques commonly used in speech coding, as discussed in Section 2.1. Several procedures, as implemented in the *auto-correlation, covariance, recursive least squares* algorithms, can be used to provide estimates for the AR coefficients of the speech model. The Recursive Least Squares (RLS) based technique is used in the proposed speech coder. We describe the RLS algorithm in Section 2.2. However, the RLS algorithm requires computations of the inverse of the auto-correlation matrix of the input data, resulting in computational complexity on the order of the square of the order of the filter [1].

In speech the formants are widely separated in frequency, they occur in the neighborhood of [400 1000 1600 2400] Hz, and consequently the corresponding pairs of poles are dominant in only certain frequency bands. This nature of speech makes it feasible to model speech as if it were generated by cascaded AR sections of lower (here, second) order [4]. The Cascaded RLS with Subsection Adaptation (CRLS-SA) algorithm [4] for adapting the AR filter coefficients associated with the cascaded model is one of the techniques, based on the RLS method, which significantly reduces the required computational effort relative to the RLS algorithm as explained in Section 2.3. The CRLS-SA takes advantage of the fact that, for inverse filtering applications, the gradients of each section in the cascade are almost uncorrelated with the gradients in other sections. In CRLS-SA the gradient auto-correlation matrix is assumed to be block diagonal, involving only 2×2 gradient auto-correlation matrices. This assumption reduces the computational complexity of RLS.

2.1 Linear Prediction

We assume that the speech signal y_n is an N^{th} order AR process represented by

$$y_n = \sum_{k=1}^{N} a_{n,k} y_{n-k} + u_n \tag{2-1}$$

where u_n is the white noise excitation of the AR model, with its coefficient vector $\tilde{\mathbf{a}}_n$, given by

$$\begin{aligned} \tilde{\mathbf{a}}_n &= \begin{bmatrix} 1 & -a_{n,1} & -a_{n,2} & \dots & -a_{n,N} \end{bmatrix}^T \\ &= \begin{bmatrix} 1 & -\mathbf{a}_n^T \end{bmatrix}^T \end{aligned} \tag{2-2}$$

$$\mathbf{a}_n = \begin{bmatrix} a_{n,1} & a_{n,2} & \dots & a_{n,N} \end{bmatrix}^T \tag{2-3}$$

and y_n is the speech sample. The subscript n, in all the parameters represent time index. Let $\mathbf{y}_n$ be a speech input vector with past values given by

$$\mathbf{y}_n = \begin{bmatrix} y_{n-1} & y_{n-2} & \dots & y_{n-N} \end{bmatrix}^T \tag{2-4}$$

Linear Prediction (LP) is a method of predicting the unknown signal from its past. Say y_n is the output of an unknown system with some unknown input u_n. Thus, from (2-1) given the past outputs, the output y_n is estimated. Let $\hat{y}_n$, be the estimated value of y_n at time n, such that

$$\hat{y}_n = \sum_{k=1}^{N} \hat{a}_{n,k} y_{n-k} \tag{2-5}$$

where

$$\hat{\mathbf{a}}_n = \begin{bmatrix} \hat{a}_{n,1} & \hat{a}_{n,2} & \dots & \hat{a}_{n,N} \end{bmatrix}^T \tag{2-6}$$

are the estimated LPC coefficients defining $\hat{A}_n(z)$:

$$\hat{A}_n(z) = \sum_{k=1}^{N} \hat{a}_{n,k} z^{-k} . \tag{2-7}$$

Since y_n is estimated based on known knowledge at time (n-1), this operation is termed *forward linear prediction* and depicted in Figure 2-1. The *'forward prediction error f_e'* is the difference between the input sample y_n and its predicted value $\hat{y}_n$, and formulated as

$$f_e = e_n = y_n - \hat{y}_n \tag{2-8}$$

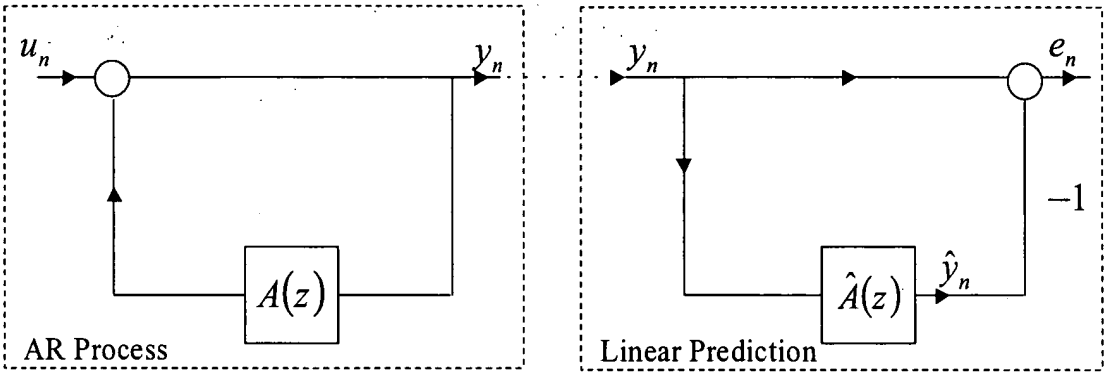

Figure 2-1 AR Process and Linear Prediction.

In other words, the linear prediction process finds an estimate of $A_n(z)$, $\hat{A}_n(z)$, by way of filtering y_n with the inverse filter $\left[1 - \hat{A}_n(z)\right]$ and minimizing the error according to some criterion. The operation of prediction error filtering applied to a stationary process $\{y_n\}$ is termed as *analysis* process. Hereby, we define an N^{th} prediction error filter, also termed analysis filter $H_a(z)$ as

$$H_a(z) = \left[1 - \hat{A}_n(z)\right] \tag{2-9}$$

where $\hat{A}_n(z)$ is as defined in (2-7). In theory, as the order of the prediction error filter increases, the correlation between input samples is reduced and eventually when the order becomes high enough, the output process consists of a sequence of uncorrelated samples [1]. A sequence of uncorrelated

random variables is called *white-noise.* A white-noise process denoted by $\{w_n\}$ has zero-mean and variance σ_w^2, such that

$$\mathrm{E}\left[w_i w_j^*\right] = \begin{cases} \sigma_w^2, & i = j \\ 0, & i \neq j \end{cases} \tag{2-10}$$

The process of converting a correlated input into white-noise output is termed as *whitening process.* When the output becomes white, the input process $\{y_n\}$ can be represented by the analysis filter coefficients and the prediction error power $P_M = \sigma_w^2$.

The auto-regressive modeling of the stationary process $\{y_n\}$, is termed as *synthesis* process. The analysis process and synthesis process are complementary to each other. In other words with white noise process $\{w_n\}$ of zero-mean and variance σ_w^2 at the input, an inverse analysis filter produces the stationary process $\{y_n\}$. The inverse analysis filter is termed as synthesis filter $H_s(z)$

$$H_s(z) = \left[1 - \hat{A}_n(z)\right]^{-1} \tag{2-11}$$

The analysis filter is an all-zero filter with an impulse response of finite duration. Conversely, the synthesis filter is an all-pole filter with an impulse response of infinite duration. The zeros of analysis filter lie inside the unit-circle and are located at exactly the same position as the poles of the synthesis filter. This ensures that the analysis filter and the inverse analysis filter or synthesis filter are both stable. Thus, the filters exhibit minimum-phase property.

In summary, given an AR speech model of order *N*, LPC analysis minimizes the residual error e_n, resulting in the filter $\left[1 - \hat{A}_n(z)\right]^{-1}$ given by

$$e_n = y_n - \left(\hat{a}_{n,1} y_{n-1} + \hat{a}_{n,2} y_{n-2} + \ldots + \hat{a}_{n,n} y_0\right) \tag{2-12}$$

where $\left[\hat{a}_{n,1} \quad \hat{a}_{n,2} \quad \ldots \quad \hat{a}_{n,N}\right]$ are the estimated LPC parameters.

Exhibit II
Ser. No. 09/749,782
Art Unit 2654
Exc D. Storm

CSE 291 LECTURE NOTES

October 30, 2002

ANNOUNCEMENTS

I'm handing out a sample solution for the first assignment (thanks Bianca!) and the new third assignment. This assignment is more open-ended and closer to how you might formulate a research project.

Today's notes are based on Chapter 3 of Hastie/Tibshirani/Friedman. They use facts from linear algebra which are not discussed explicitly in the chapter. We'll try to state these during the lecture.

LINEAR REGRESSION

Suppose we want to learn how a random variable Y depends on random variables X1 ... Xp. This is a multivariate scenario unlike the univariate situations we've seen so far. The model we assume is linear:

$$E[Y|X] = f(X) = b0 + SUM_j Xj * bj$$

The parameters we want to estimate (also called coefficients) are the p+1 scalars bj.

Suppose we have N training examples (subscript i). Each training example is a p-dimensional column vector (subscript j) along with a scalar. The components of a training example are called independent variables, features, attributes, or predictors. These are all synonyms!

The most popular estimation method is called "least squares." We pick the b vector to minimize

$$RSS(b) = SUM_i (f(xi) - yi)^2$$

"RSS" stands for "residual sum of squares." Note that this minimization treats all the xi equally, and that it penalizes large deviations dramatically. Background knowledge about the real-world scenario may imply that these choices are not appropriate. But we will see soon that least squares gives minimum-variance unbiased estimates.

For now let's look at least squares algorithmically and geometrically.

MINIMIZING RSS

Let X be a matrix where each row is one training example, and the first column is all ones, so the size of X is N by p+1. Let y be the column vector of observed values for the dependent variable. In matrix notation

$$RSS(b) = (y - Xb)^T(y - Xb)$$

To explain this in detail: Xb is a matrix times a column vector, so we take the dot-product of each row of X with b. This gives a column vector of size N, which we can subtract from y, giving e = y-Xb. Now we take the dot product of e with itself by first converting it into a row vector with the transpose operation, indicated by the superscript T.

Note that y and X are fixed and the result RSS(b) is a scalar function of the p+1 parameters that are components of the vector b. To minimize RSS(b) we set its derivative to zero. The derivative is

$$d/db\ RSS(b) = -2X^T(y - Xb)$$

This result can be proved by going back to the non-matrix formulation and computing the derivative of the RSS sum using standard calculus. The second derivative is

$d/db\ -2XT(y - Xb) = -2X^TX$

We can solve the equation $-2X^T(y - Xb) = 0$ using the matrix inverse:

$$2X^Ty = -2X^T(-Xb)$$
$$(X^TX)^{-1}X^Ty = b$$

This solution is only valid if the inverse actually exists, i.e. the matrix X^TX is non-singular. Note that X^TX is square,of size p+1 by p+1.

We can also prove that any solution of $-2X^T(y - Xb) = 0$ minimizes the RSS using linear algebra; see Appendix A of Silvey.

LINEAR DEPENDENCE

Suppose the columns of X are not linearly independent, e.g. one feature is a linear ombination of other features. This will happen for example if we use a "one of n" coding for a feature that i sintrinsically discrete. Then X^TX is singular and the least-squares coefficients b are not defined uniquely. This makes sense: you can get equally good predictions for y from alternative linear functions of the input vector. But the predicted values y hat are still uniquely defined.

Linear dependence between columns will also happen when the number of rows (i.e. training examples) is less than the number of columns (i.e. features)
